Búsqueda interna

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

Introducción

- Los computadores emplean gran parte de su tiempo en operaciones de búsqueda y ordenamiento.
- En la búsqueda al igual que en el ordenamiento existen 2 casos típicos de búsqueda:
  - búsqueda interna (de arrays)
  - búsqueda externa (archivos)
- Los arrays se almacenan en la memoria interna o central, de acceso aleatorio y directo, y por ello su gestión es rápida.

147

Introducción

- Los archivos se sitúan adecuadamente en dispositivos de almacenamiento externo, que son más lentos y basados en dispositivos mecánicos: cintas y discos magnéticos.

148

Búsqueda secuencial o lineal

- Se explora el vector en forma secuencial, es decir, se recorre el vector desde el primer elemento al último.
- Se compara cada elemento del vector con el valor deseado
- Esta comparación se detiene cuando el elemento es encontrado o se recorrió todo el vector.
- No requiere vector ordenado.

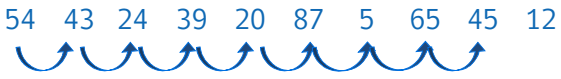
149

Búsqueda secuencial o lineal

```
void busqueda_secuencial(int v[20],int valor)
{
  int i,enc=0;
  while ((!enc)&&(i<20)){
    if (v[i]==valor) enc=1;
    else i++;
  }
  if (enc) printf("valor %d encontrado en posición %d",valor,i);
  else printf("valor no encontrado");
}
```


Búsqueda secuencial o lineal

— Ejemplo: buscar número 45




```
void busqueda_secuencial(int v[20],int valor){
  int i,enc=0;
  while ((!enc)&&(i<20)){
    if (v[i]==valor) enc=1;
    else i++;
  }
  if (enc) printf("valor %d encontrado en posición %d",valor,i);
  else printf("valor no encontrado");
}
```

Búsqueda binaria

- Suponga que desea buscar el nombre de una persona en una guía telefónica. Normalmente no se buscaría en forma secuencial, sino que se buscaría en la primera o segunda mitad de la guía, una vez en esa mitad, se volvería a tantear en una de sus dos sub-mitades, y así sucesivamente se repetiría el proceso.
- Requiere vector ordenado

5 14 24 39 43 45 56 65 71 87

152

Búsqueda binaria

- Ejemplo

```
void busq_bin(int v[20],int valor) {
  int p,u,c,enc=0;
  p=0;u=9;
  while ((p<=u)&&(!enc)) {
    c=(p+u)/2;
    if (v[c]==valor) enc=1;
    else {
      if (v[c]>valor) u=c-1;
      else p=c+1;
    }
  }
  if (enc) printf("valor %d encontrado
              en posición %d",valor,i);
  else printf("valor no encontrado");
}
```

[5 14 24 39 **43** 45 56 65 71 87]

5 14 24 39 43 [45 56 **65** 71 87]

5 14 24 39 43 [**45** 56] 65 71 87

153

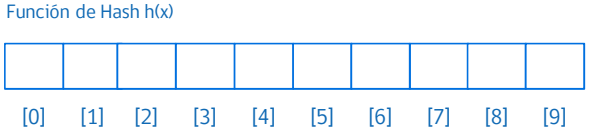
Búsqueda mediante hash table

- La búsqueda binaria reduce el tiempo requerido para buscar en una lista, sin embargo necesita los datos de manera ordenada.
- El método de transformación de claves (hashing) permite buscar de manera rápida sobre datos no ordenados.

154

Búsqueda mediante hash table

- Se compone de 2 partes:
  - Un arreglo con los datos a ser guardados y
  - Una función de hash, que convierte una clave dada (número o string) en una dirección (índice).



155

Búsqueda mediante hash table

- Ejemplo 1. Se desean guardar números de 2 cifras (0-99).
  - Función de Hash h(x) = Truncar x a la primera cifra.
  - Es decir, h(77) = 7, h(42) = 2, h(35) = 5, etc.
  - Entonces agregando a la HashTable los valores 77, 42 y 35 queda:

		42			35		77		
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

156

Búsqueda mediante hash table

- ¿Qué sucede con este caso?

							37		
		42			35		77		
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

157

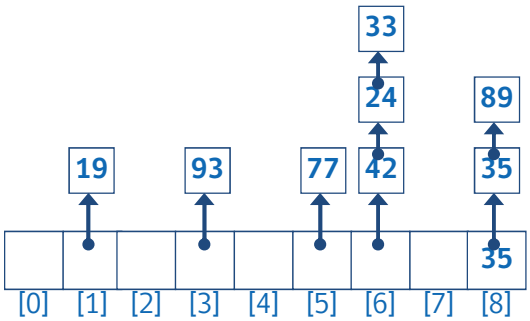
Búsqueda mediante hash table

- Qué hacer con el nuevo elemento?
  - Cambiar la estructura del arreglo de modo que pueda almacenar más de 1 elemento en la misma posición. Alternativas:
  - un arreglo de arreglos
  - un arreglo de listas

158

Búsqueda mediante hash table

- En el caso anterior con h(x) = x mod 9, sería:



159

Ordenamiento

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

Ordenamiento interno

- Los computadores emplean gran parte de su tiempo en operaciones de búsqueda y ordenamiento.
- Existen 2 métodos de ordenación: ordenación interna (de arrays) y ordenación externa (archivos).
- Los arrays se almacenan en la memoria interna o central, de acceso aleatorio y directo, y por ello su gestión es rápida.

Ordenamiento interno

— Los archivos se sitúan adecuadamente en dispositivos de almacenamiento externo, que son más lentos y basados en dispositivos mecánicos: cintas y discos magnéticos.

Método de intercambio o burbuja

— Se basa en el principio de comparar pares de elementos adyacentes e intercambiarlos entre si hasta que estén todos ordenados.

```
void burbuja1(int n,float a[max]) {
  int i,j;
  float aux;
  for (i=0;i<=n-1;i++)
    for (j=0;j<=n-1;j++)
      if (a[j]>a[j+1]) {
        aux=a[j];
        a[j]=a[j+1];
        a[j+1]=aux;
      }
}
```

Método de intercambio o burbuja

— Ejemplo: Consideremos ordenar el siguiente arreglo con 10 enteros.

72 64 50 23 84 18 37 99 45 8

72	**64**	50	23	84	18	37	99	45	8
64	**72**	**50**	23	84	18	37	99	45	8
64	50	**72**	**23**	84	18	37	99	45	8
64	50	23	**72**	**84**	18	37	99	45	8
64	50	23	72	**84**	**18**	37	99	45	8
64	50	23	72	18	**84**	**37**	99	45	8
64	50	23	72	18	37	**84**	**99**	45	8
64	50	23	72	18	37	84	**99**	**45**	8
64	50	23	72	18	37	84	45	**99**	**8**
64	**50**	23	72	18	37	84	45	8	99
50	**64**	**23**	72	18	37	84	45	8	99
50	23	**64**	**72**	18	37	84	45	8	99
50	23	64	**72**	**18**	37	84	45	8	99
50	23	64	18	**72**	**37**	84	45	8	99
50	23	64	18	37	**72**	**84**	45	8	99
50	23	64	18	37	72	**84**	**45**	8	99
50	23	64	18	37	72	45	**84**	**8**	99
50	23	64	18	37	72	45	8	**84**	**99**

164

Burbuja mejorado

— Se puede realizar una mejora en la velocidad de ejecución del algoritmo.

— En el primer recorrido del vector (cuando I=1) el valor mayor del vector se mueve al último elemento a[n]. Por lo tanto, en el siguiente paso no es necesario comparar a[n-1] y a[n].

— Esto se refleja decrementando en 1 el límite superior del segundo bucle for después de cada paso.

165

Burbuja mejorado

```
void burbuja2(int n,float a[max]) {
  int i,j;
  float aux;
  for (i=0;i<=n-1;i++)
    for (j=0;j<=n-1-i;j++)
      if (a[j]>a[j+1]) {
        aux=a[j];
        a[j]=a[j+1];
        a[j+1]=aux;
      }
}
```

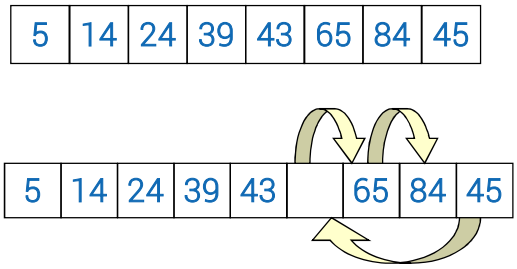
166

Método baraja o por inserción directa

- Consiste en insertar un elemento en el vector en una parte ya ordenada de este vector y comenzar de nuevo con los elementos restantes.
- Esta inserción se realiza más fácilmente con una bandera (sw).

167

Método baraja o por inserción directa




```
void ins_directa(int n,float a[max]) {
  int i,j,sw;
  float aux;
  for (i=1;i<=n-1;i++) {
    aux=a[i];
    j=i-1;
    sw=0;
    while((j>=0) && (sw==0))
      if (a[j]>aux){
        a[j+1]=a[j];
        j--;
      } else sw=1;
    a[j+1]=aux;
  }
}
```


Método de inserción binaria

— El método de inserción directa se puede mejorar fácilmente realizando, para ubicar el lugar de inserción, una búsqueda binaria en vez de una búsqueda secuencial.

```
void insercion_binaria(int n,float a[max]){
 int i,j,p,u,c;
 float aux;
 for (i=1;i<n;i++){
   aux=a[i];
   p=0;
   u=i-1;
   while (p<=u){
     c=(p+u)/2;
     if (a[c]>aux)  u=c-1;
     else p=c+1;
   }
   for (j=i-1;j>=p,j--)
     a[j+1]=a[j];
   a[p]=aux;
 }
}
```

Método por selección

- Este método se basa en buscar el elemento menor del vector y ubicarlo en primera posición.
- Luego se busca el segundo elemento más pequeño y se ubica en la segunda posición, y así sucesivamente.

```
void seleccion(int n,float a[max]) {
  int i,j,k;
  float aux;
  for (i=0;i<n-1;i++) {
    k=i;
    for (j=i+1;j<n;j++)
      if (a[j]<a[k]) k=j;
    aux=a[k];
    a[k]=a[i];
    a[i]=aux;
  }
}
```

170

Método Shell

- Es una mejora del método de inserción directa que se utiliza cuando el número de elementos a ordenar es grande.
- En el método de ordenación por inserción, cada elemento se compara con los elementos contiguos de su izquierda, uno tras otro. Si el elemento a insertar es uno de los más pequeños, hay que ejecutar muchas comparaciones antes de ubicarlo en su lugar correcto.
- Shell modificó los saltos contiguos resultantes de las comparaciones por saltos de mayor tamaño y con eso se consigue una ordenación más rápida.

171

Método Shell

```
void shell(int n,float a[max]) {
  int i,j,k,salto;
  float aux;
  salto = n/2;
  while (salto>0) {
    for (i=salto;i<n;i++) {
      j=i-salto;
      while (j>=0) {
        k=j+salto;
        if (a[j]<=a[k]) j=-1;
        else {
          aux=a[j];
          a[j]=a[k];
          a[k]=aux;
          j=j-salto;
        }
      }
    }
    salto=salto/2;
  }
}
```

172

Especificación sobre árboles

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

1

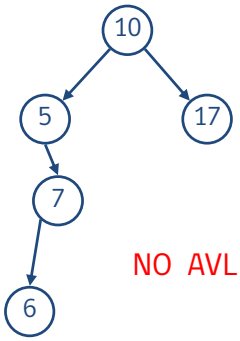
AVL

— Se dice que un árbol binario está balanceado si y sólo si en cada nodo las alturas de sus 2 subárboles difieren como máximo en 1.

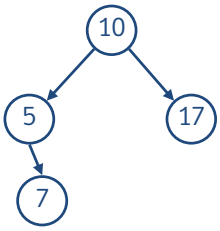
— Todos los árboles perfectamente balanceados son árboles AVL.

2

AVL



NO AVL



AVL

3

3

AVL: rotaciones

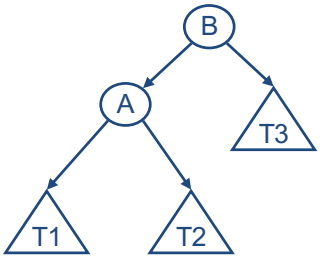
— A través de rotaciones es posible transformar un árbol binario cualquiera en un árbol AVL.

4

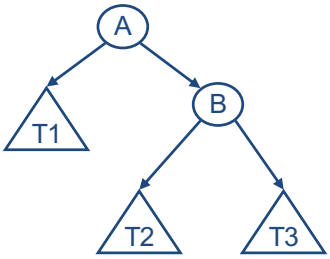
4

AVL: rotaciones

— Rotación derecha simple:



((T1 A T2) B T3)



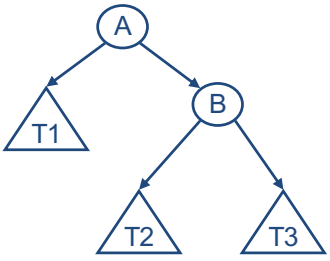
(T1 A (T2 B T3))

5

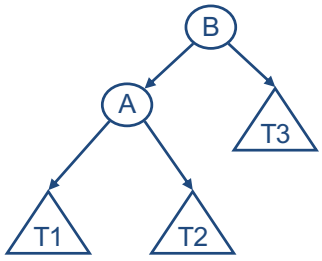
5

AVL: rotaciones

— Rotación izquierda simple:



(T1 A (T2 B T3))



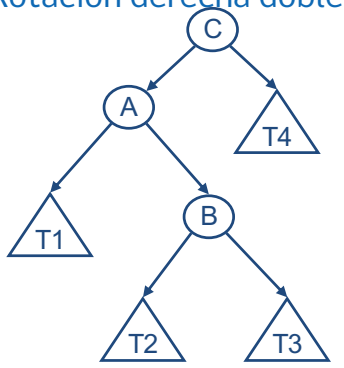
((T1 A T2) B T3)

6

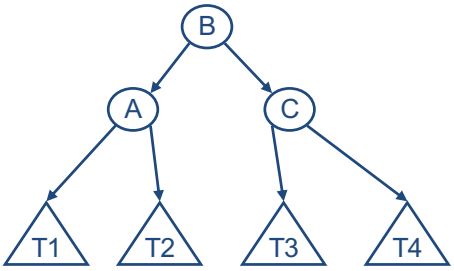
6

AVL: rotaciones

— Rotación derecha doble:



(T1 A (T2 B T3) C T4)



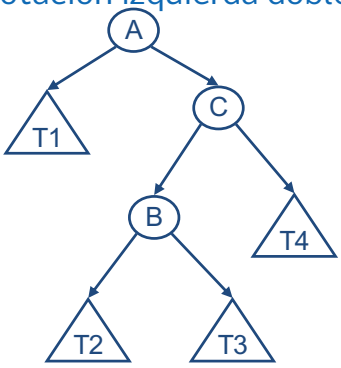
((T1 A T2) B (T3 C T4))

7

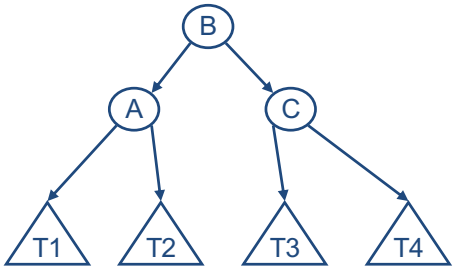
7

AVL: rotaciones

— Rotación izquierda doble:



(T1 A (T2 B T3) C T4)



((T1 A T2) B (T3 C T4))

8

8

Representación de árboles binarios

- Los árboles binarios (y también de búsqueda - ABB) pueden ser representados de distintas formas, utilizando:
  - Punteros
  - Tres vectores
  - Un vector de valores
  - Un vector de estructuras

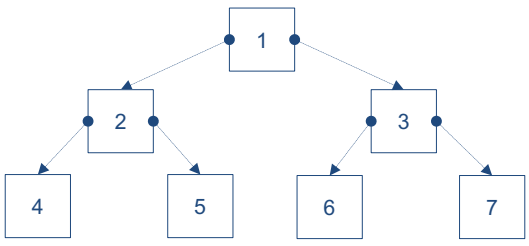
9

9

Representación de árboles binarios

- Ejemplo, mediante punteros:

```
struct nodo
{
    int elem;
    struct nodo *izq, *der;
};
```



10

10

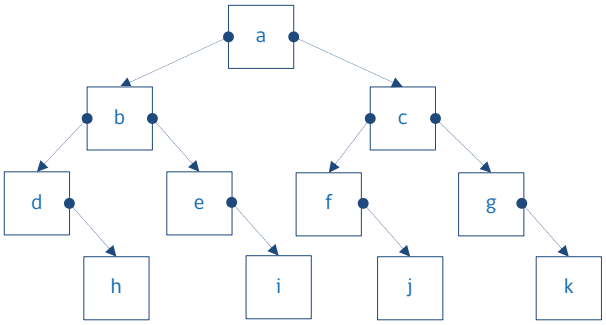
Representación de AB mediante 3 vectores

— Esta alternativa considera 3 arreglos lineales paralelos, que contemplan el campo de información y las posiciones de los 2 subárboles.

11

11

Representación de AB mediante 3 vectores



	INFO.	IZQ	DER
1	a	4	11
2	g	0	5
3	d	0	8
4	b	3	7
5	k	0	0
6	f	0	9
7	e	0	10
8	h	0	0
9	j	0	0
10	i	0	0
11	c	6	2
...			
N			

12

12

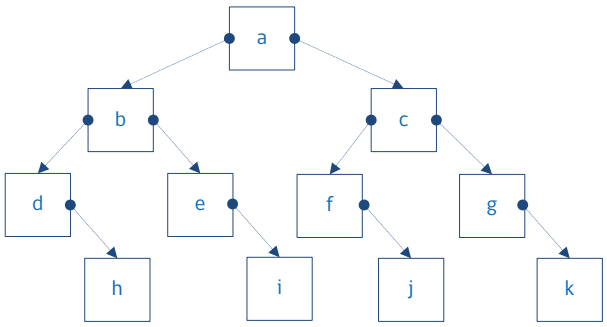
Representación de AB mediante vector de valores

- Se considera un solo arreglo unidimensional con la información requerida (llamado ARBOL).
- Para un árbol de profundidad h, se necesita un arreglo de largo 2h+2.
- La raíz se guarda en ARBOL[1]
- Si un nodo n está en ARBOL[i] entonces:
  - su hijo izquierdo va en ARBOL[2*i]
  - su hijo derecho va en ARBOL[2*i+1]

13

13

Representación de AB mediante vector de valores



1	a
2	b
3	c
4	d
5	e
6	f
7	g
8	
9	h
10	
11	i
12	
13	j
14	
15	k
...	

14

14

Limitaciones de los árboles binarios

- A menudo se usan árboles binarios de búsqueda para ordenar listas de valores, minimizando el número de lecturas, y evitando tener que ordenar dichas listas. Pero este tipo de árboles tienen varias desventajas:
  - Es difícil construir un árbol binario de búsqueda perfectamente equilibrado.
  - El número de consultas en el árbol no equilibrado es impredecible.
  - Y además el número de consultas aumenta rápidamente con el número de registros a ordenar.

15

15

Limitaciones de los árboles binarios

- Para evitar estos inconvenientes se usan árboles-B, sobre todo cuando se ordenan ficheros, donde se ha convertido en el sistema de indexación más utilizado.

16

16

Árboles B

- Algoritmo desarrollado en 1972 por por Rudolf Bayer y Eduard M. McCreight.
- La letra B no significa "binario", ya que:
  - Los árboles-B nunca son binarios.
  - Y tampoco es porque sean árboles de búsqueda, ya que en inglés se denominan B-trees.
  - Tampoco es porque sean balanceados, ya que no suelen serlo.

17

17

Árboles B

- Los árboles B son índices multi-nivel que resuelven los problemas de la inserción y borrado de registros y se basan en dos reglas:
  - Permitir que los registros índice no estén llenos
  - No mover índices de un registro a otro cuando está lleno. En su lugar, dividir en dos el registro. Cuando dos registros están muy vacíos, unirlos en uno solo.

18

18

Árboles B

— Las características que debe cumplir un árbol-B son:

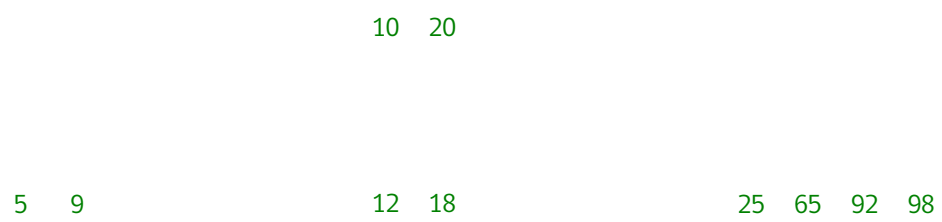
- Un parámetro muy importante en los árboles-B es el ORDEN (m). El orden de un árbol-B es el número máximo de claves (valores) que puede almacenar un nodo.
- Si n es el número de claves presentes en un nodo de un árbol-b, dicho nodo debe contener n+1 ramas (o enlaces a otros nodos).
- El árbol está ordenado.
- Todos los nodos terminales (nodos hoja), están en el mismo nivel.
- El número máximo de claves por nodo es m.
- El número mínimo de claves por nodo es m/2, excepto la raíz
- La profundidad (h) es el número máximo de consultas para encontrar una clave.

19

19

Árboles B

— Ejemplo de un árbol B de orden 4 y profundidad 2:



20

20

Árboles B+

— Los árboles B+ constituyen otra mejora sobre los árboles B, pues conservan la propiedad de acceso aleatorio rápido y permiten además un recorrido secuencial rápido.

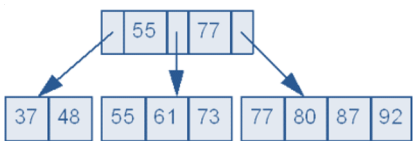
— En un árbol B+ todas las claves se encuentran en hojas, duplicándose en la raíz y nodos interiores aquellas que resulten necesarias para definir los caminos de búsqueda.

21

21

Árboles B+

— Para facilitar el recorrido secuencial rápido las hojas se pueden vincular, obteniéndose de esta forma una trayectoria secuencial para recorrer las claves del árbol.



22

22

Teoría de grafos

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

23

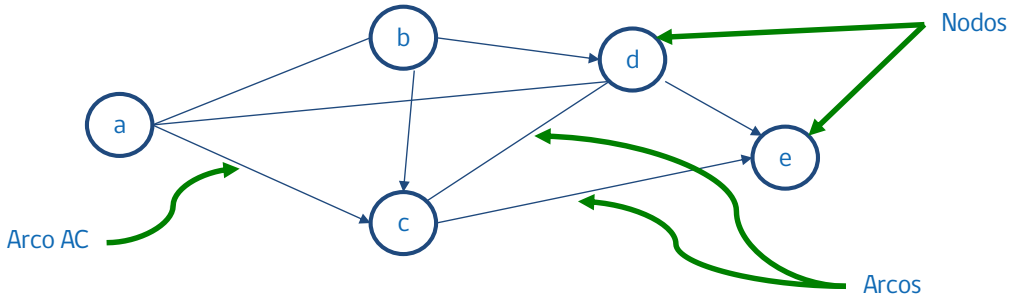
Definición general

- Un grafo es un conjunto de puntos y un conjunto de líneas llamadas aristas o arcos, cada una de las cuales une un punto llamado nodo o vértice con otro.
- Se representan el conjunto de vértices de un grafo dado G por VG = {a,b,c,d,e}
- El conjunto de arcos por AG = {ab, ac, ad, bd, bc, cd, ce, de}

24

24

Definición general



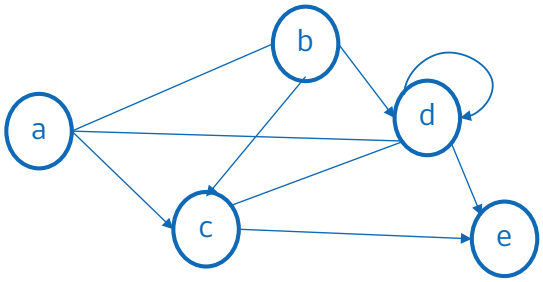
25

25

Términos importantes

— Una arista se representa por los vértices que conecta. La arista que conecta los vértices b y d se representa por A(b,d).

— Algunos vértices pueden conectar un nodo consigo mismo; por ejemplo, el vértice d tiene el formato A(d,d). Estas aristas se denominan bucles.

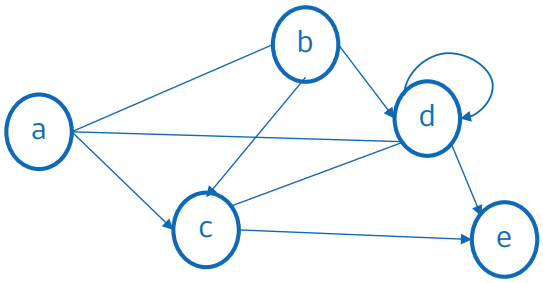


26

26

Términos importantes

- Al número de vértices que tiene un grafo se le llama orden del grafo
- Un grafo nulo es un grafo de orden 0.
- Dos vértices son adyacentes si hay un arco que los une.

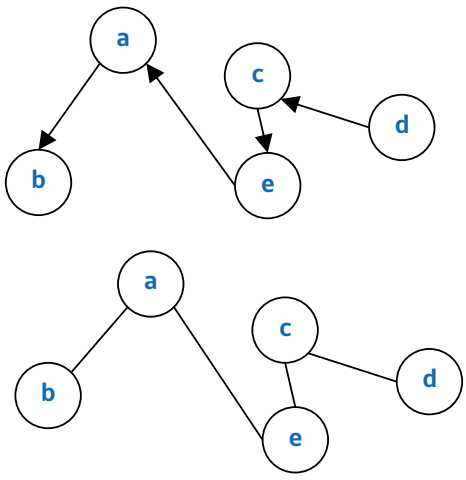


27

27

Términos importantes

- Un camino o trayectoria entre 2 nodos es una sucesión de arcos distintos que conectan estos nodos. En el grafo anterior, una trayectoria que conecta al nodo a con el nodo e es ab-bd-de.
- La longitud de un camino es el nº de arcos que comprende.

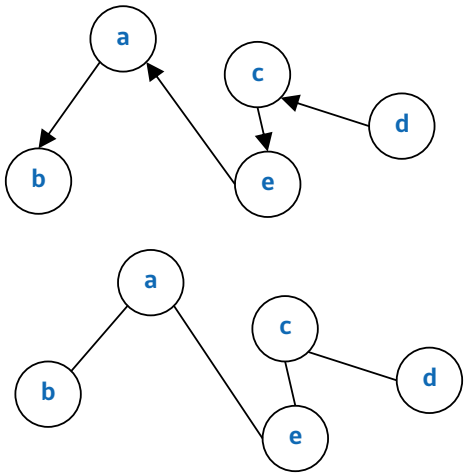


28

28

Términos importantes

- Un grafo es dirigido cuando los arcos tienen dirección.
- Un grafo es no-dirigido cuando los arcos no tienen dirección.

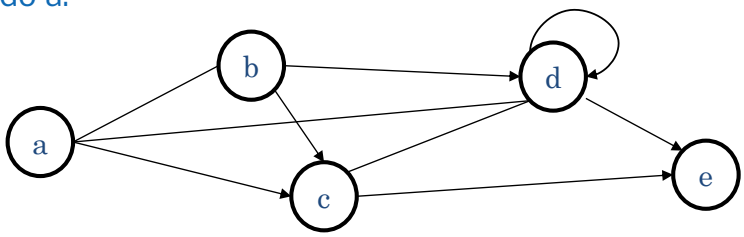


29

29

Términos importantes

- Un camino o trayectoria no dirigida del nodo i al nodo j es una sucesión de arcos cuya dirección (si la tienen) puede ser hacia o desde el nodo j. De lo anterior se desprende que una trayectoria dirigida también es no dirigida, pero no a la inversa. Ej. de-bd-ba es una trayectoria no dirigida del nodo e hacia el nodo a.

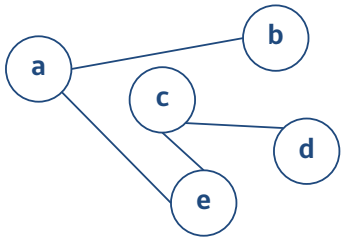


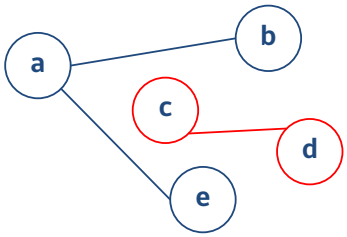
30

30

Términos importantes

— Un grafo se denomina conectado cuando existe siempre un camino que une 2 vértices cualquiera y desconectado en caso contrario



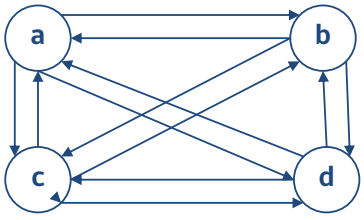


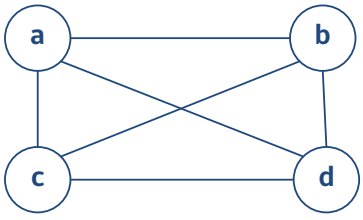
31

31

Términos importantes

— Un grafo es completo cuando cada vértice está conectado con todos y cada uno de los nodos restantes (cada nodo es adyacente a los demás)





32

32

Términos importantes

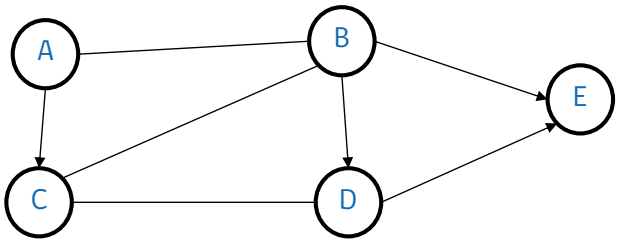
- Un ciclo es una trayectoria que comienza y termina en el mismo nodo.
- En una red dirigida, un ciclo puede ser dirigido o no dirigido, según si la trayectoria en cuestión es dirigida o no dirigida.
- Como una trayectoria dirigida también es no dirigida, entonces un ciclo dirigido también es no dirigido (pero al revés no siempre es cierto).

33

33

Términos importantes

- En la red de abajo, el ciclo AC-CB-BA es dirigido, el ciclo BD-DE-BE es no dirigido:



34

34

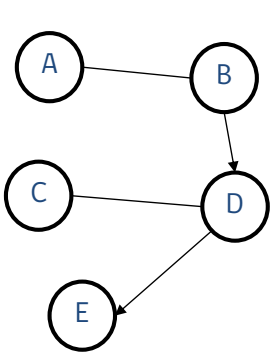
Términos importantes

— Un **árbol de expansión** es una red conexa que no contiene ciclos no dirigidos. Consiste en N nodos con N-1 arcos.

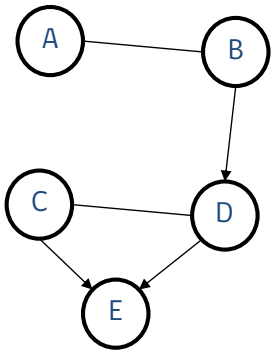
35

35

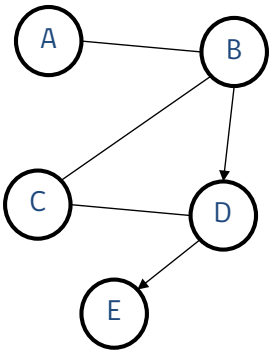
Términos importantes



Sí es árbol
de expansión



No es árbol
de expansión



No es árbol
de expansión

36

36

Términos importantes

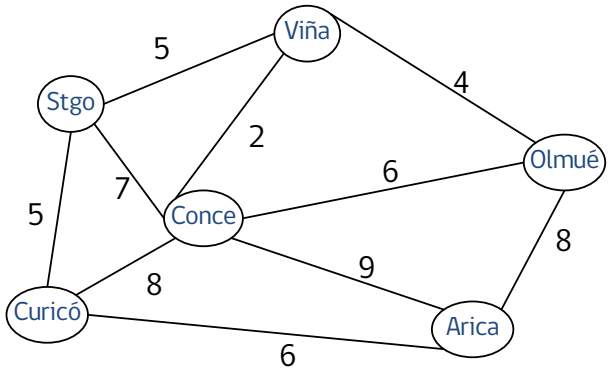
- La cantidad máxima de flujo que puede circular en un arco dirigido es llamada capacidad del arco.
- El nodo que tiene la propiedad de que su flujo que sale es mayor que el flujo que entra en él se le llama nodo fuente (o nodo orígen).
- Por el contrario, si su flujo que sale es menor que el flujo que entra a él se le llama nodo demanda (o nodo destino).
- Si el flujo que entra es igual al flujo que sale, entonces se le llama nodo de trasbordo (o nodo intermedio)

37

37

Términos importantes

- Un grafo es etiquetado cuando cada arco y/o vértice tienen algún tipo de rótulo, por ejemplo un nombre, un costo o algún otro valor.



38

38

Representación de grafos

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

39

Matriz de adyacencia

- Existen dos técnicas estándar para representar un grafo G:
  - la matriz de adyacencia (mediante arrays) y
  - la lista de adyacencia (mediante punteros/listas enlazadas).
- La matriz de adyacencia M es un array de dos dimensiones que representa las conexiones entre sus vértices.

$$M(i,j) \begin{cases} 1 \text{ si existe una arista } (V_i, V_j) \text{ en } A_g, V_i \text{ es adyacente a } V_j \\ 0, \text{ en caso contrario} \end{cases}$$

40

40

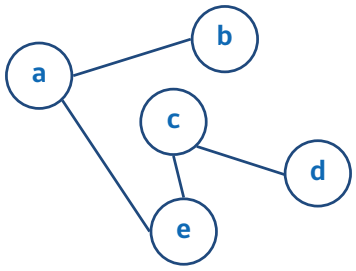
Matriz de adyacencia

— Las columnas y las filas de la matriz representan los vértices del grafo.

— Si G es un grafo no dirigido, la matriz es simétrica M(i,j) = M(j,i).

— Ejemplo 1

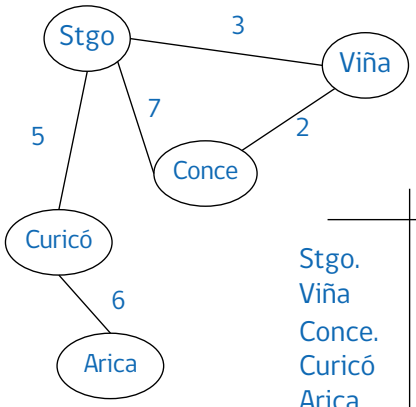
	a	b	c	d	e
a	0	1	0	0	1
b	1	0	0	0	0
c	0	0	0	1	1
d	0	0	1	0	0
e	1	0	1	0	0



41

41

Matriz de adyacencia



	Stgo	Viña	Conce	Curicó	Arica
Stgo.	-	3	7	5	-
Viña	3	-	2	-	-
Conce.	7	2	-	-	-
Curicó	5	-	-	-	6
Arica	-	-	-	6	-

42

42

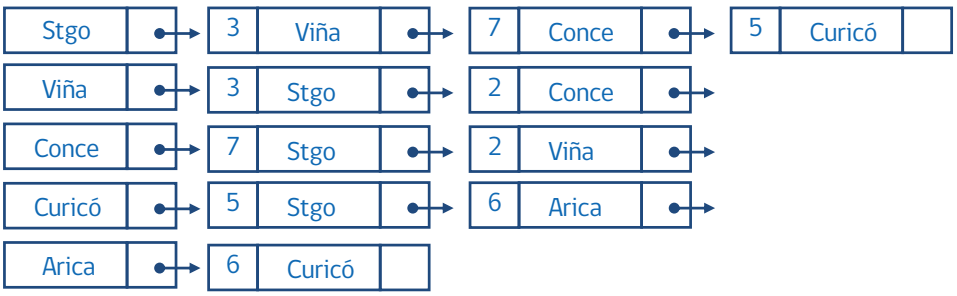
Lista de adyacencia

- La lista de adyacencia es útil cuando un grafo tiene muchos vértices y pocas aristas.
- Esta lista está compuesta por dos partes: un directorio y un conjunto de listas enlazadas.
- Hay una entrada en el directorio por cada nodo del grafo. La entrada en el directorio del nodo i apunta a una lista enlazada que representa los nodos que son conectados al nodo i.

43

43

Lista de adyacencia



44

44

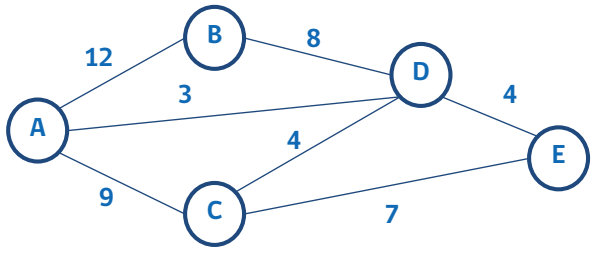
Cobertura o expansión mínima

INF2240 – Estructura de datos
Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

45

Cobertura o expansión mínima

— Este problema considera una red no dirigida y conexa. En ella se debe encontrar un árbol de expansión con la longitud total mínima de sus ligaduras.



46

46

Cobertura o expansión mínima

- En la práctica se puede utilizar para:
  - minimizar la cantidad de pavimento para conectar entre sí un grupo de ciudades,
  - minimizar la cantidad de cable a utilizar para unir una serie de enchufes,
  - minimizar la cantidad de tubería a utilizar para la instalación de alcantarillado a una población, etc.
  - minimizar el cableado para una red.

47

47

Cobertura o expansión mínima

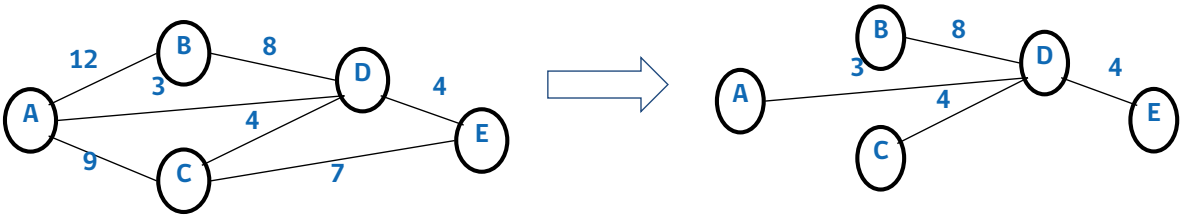
- El algoritmo de Prim permite encontrar una solución óptima a este problema:
  - Se selecciona arbitrariamente cualquier nodo y se conecta al nodo más cercano distinto de éste.
  - Se identifica el nodo no conectado más cercano a un nodo conectado y se conectan mediante una ligadura.
  - Si quedan nodos sin conectar se vuelve al paso 2. Los empates se solucionan arbitrariamente.

48

48

Cobertura o expansión mínima

— Ejemplo 1



49

49

Ejemplo práctico

— Se desea construir una red de caminos que permita conectar a siete ciudades. A continuación se presentan las distancias (km) entre cada par de ciudades. Las distancias que aparecen en blanco indican que no es posible técnicamente construir un camino directo entre ese par de ciudades.

50

50

Ejemplo práctico

Ciudad	O	A	B	C	D	E	T
O	0	3	5	5			
A	3	0	1		7		
B	5	1	0	1	3	3	
C	5		1	0		4	
D		7	3		0	1	4
E			3	4	1	0	6
T					4	6	0

51

51

Ejemplo práctico

— El costo del pavimento es de 1000 M$ por kilómetro. Ayude a la empresa concesionaria a diseñar una red de caminos, conectando las ciudades a un mínimo costo total.

— Su representación mediante una red queda:

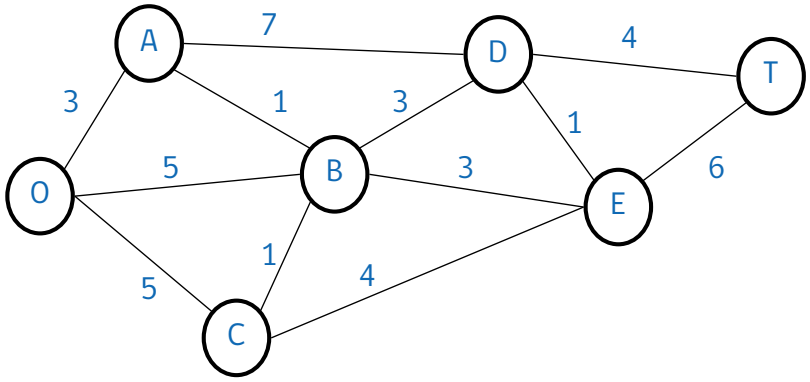
→

52

52

Ejemplo práctico

— Su representación mediante una red queda:

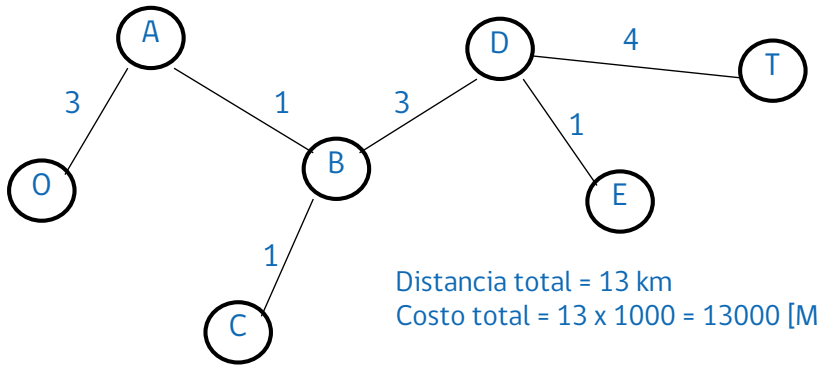


53

53

Ejemplo práctico

— Utilizando el algoritmo de Prim (partiendo desde O), el árbol de mímima expansión es:



Distancia total = 13 km
Costo total = 13 x 1000 = 13000 [M$]

54

54